## REMARKS

Applicant is in receipt of the Office Action mailed June 6, 2006. Claims 1, 3-6, 8, and 12-23, 25, 26, 28, 29, and 31-36 are pending in the case. Reconsideration of the present case is earnestly requested in light of the following remarks.

**Amendments**

Applicant has amended the application to change the title.

The new Abstract has been submitted on a separate sheet, as required by the Examiner. Applicant has amended the application to change the title to correct a typographical error.

Regarding claim 23, Applicant had mistakenly included the struck-through "programmatically" in the added text of the claim. The intended amended claim should have read as follows:

23. (Currently Amended) A computer-implemented method for ~~programmatically~~ automatically generating a new graphical program, comprising:

receiving information specifying a state diagram, wherein the state diagram specifies first functionality;

executing a graphical program generation (GPG) program;

the GPG program ~~programmatically~~ automatically generating the new graphical program using said information, wherein the new graphical program includes graphical source code corresponding to the state diagram, wherein the new graphical program comprises a plurality of interconnected nodes which visually indicate operation of the graphical program, [[and]] wherein the new graphical program is executable by a computer to perform the first functionality, and wherein said automatically generating the new graphical program creates the new graphical program without any user input specifying the new graphical program during said creating.

Applicant respectfully requests that the Examiner omit the struck-through word in the amended claim.

4

**Objections**

The Office Action objected to the Specification for use of the term "may". Applicant respectfully submits that there is nothing in the statutes, rules, or MPEP that forbids use of this term in patent applications. Applicant notes that there is no requirement that an application only describe a single embodiment of the invention, and that use of the term "may" is a commonly accepted way of referring to features that are included in some embodiments, but possibly not in others. Applicant notes that a search of patents in the USPTO database that issued in the past year alone that use the term "may" in the specification returns more than 154,000 patents.

Applicant thus respectfully submits that the objection to the Specification is improper, and earnestly requests its removal.

**Non-Statutory Double Patenting Rejections**

Claims 1, 3-6, 8, 12-23, 25, 26, 28, 29, and 31-54 were provisionally rejected for nonstatutory obviousness-type double patenting as being unpatentable over claims 285-381 of copending Application No. 09/518,492 (henceforth, "'492") in view of MathWorks ("Stateflow for State Diagram Modeling User's Guide", version 4, 1997-2001, henceforth, "MathWorks"). Applicant respectfully disagrees, and believes that these claims are patentably distinct and non-obvious over the cited art. For example, Applicant notes that '492 nowhere even mentions "state diagram", or even "state", anywhere in the application, nor indicates or even hints at the desirability of generating a graphical program from state diagram information. However, in order to expedite prosecution of the case, Applicant has included herewith a Terminal Disclaimer with respect to '492, and respectfully requests removal of the double patenting rejection.

Claims 1, 3-6, 8, 12-23, 25, 26, 28, 29, and 31-54 were rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over U.S. Patent No. 7,000,190 (henceforth, "'190") in view of MathWorks. Applicant has included herewith a Terminal Disclaimer with respect to '190, and respectfully requests removal of the double patenting rejection.

5

**Section 102 Rejections**

Claims 1, 3-6, 8, 12-23, 25, 26, 28, 29, and 31-54 were rejected under 35 U.S.C. 102(e) as being anticipated by MathWorks. Applicant respectfully disagrees.

As the Examiner is certainly aware, anticipation requires the presence in a single prior art reference disclosure of each and every element of the claimed invention, arranged as in the claim. *Lindemann Maschinenfabrik GmbH v. American Hoist & Derrick Co.*, 221 USPQ 481, 485 (Fed. Cir. 1984). The identical invention must be shown in as complete detail as is contained in the claims. *Richardson v. Suzuki Motor Co.*, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989).

Claim 1 recites:

1.     A computer-implemented method for automatically generating a graphical program based on a state diagram, comprising:

receiving state diagram information, wherein the state diagram information represents the state diagram and specifies a plurality of states;

automatically generating the graphical program in response to the state diagram information, wherein said automatically generating comprises automatically generating graphical source code corresponding to the plurality of states, wherein the graphical source code comprises a plurality of interconnected nodes which visually indicate functionality of the graphical program, wherein the graphical program is executable by a computer, and wherein said automatically generating the graphical program creates the graphical program without any user input specifying the graphical program during said creating.

In addition to the arguments provided in the Response to the previous Office Action, which is hereby incorporated by reference, Applicant presents the following reasons for the allowability of claim 1:

Nowhere does MathWorks teach or suggest **automatically generating the graphical program in response to the state diagram information, wherein said**

automatically generating comprises automatically generating graphical source code corresponding to the plurality of states, wherein the graphical source code comprises a plurality of interconnected nodes which visually indicate functionality of the graphical program, wherein the graphical program is executable by a computer, and wherein said automatically generating the graphical program creates the graphical program without any user input specifying the graphical program during said creating.

The Office Action asserts that MathWorks discloses "automatically generating the graphical program in response to the state diagram information", citing MathWorks, p.1-6, which, along with subsequent text, reads:

> Creating a Simulink Model
> Opening the Stateflow model window is the first step toward creating a Simulink model with a Stateflow block. By default, an untitled Simulink model with an untitled, empty Stateflow block is created for you when you open the Stateflow model window. You can either start with the default empty model or copy the untitled Stateflow block into any Simulink model to include a Stateflow diagram in an existing Simulink model.
>
> These steps describe how to create a Simulink model with a Stateflow block, label the Stateflow block, and save the model:
>
> 1      Display the Stateflow model window.
> At the MATLAB prompt enter stateflow.
> MATLAB displays the Stateflow block library.
>
>      The library contains an untitled Stateflow block icon, an Examples block, and a manual switch. Stateflow also displays an untitled Simulink model window with an untitled Stateflow block.
>
> 2      Label the Stateflow block.
> Label the Stateflow block in the new untitled model by clicking in the text area and replacing the text Untitled with the text On_off.
>
> 3      Save the model.
> Choose Save As from the File menu of the Simulink model window. Enter a model title.
>
>      You can also save the model by choosing Save or Save As from the Stateflow graphics editor File menu. Saving the model either from Simulink or from the graphics editor saves all contents of the Simulink model.

Applicant respectfully submits that the cited text, and MathWorks in general, fails to teach or suggest *automatically generating the graphical program in response to the state diagram information*, as claimed. Rather, the cited reference describes *manual* creation, i.e., based on user input, of a Simulink model with a Stateflow block. Subsequent portions of MathWorks, e.g., p.1-9 through 1-12, disclose *manual* creation of a Stateflow diagram, i.e., in response to user input. No mention is made of *automatically* generating a graphical program based on such state diagram information. Thus, MathWorks fails to teach or suggest this feature of claim 1.

The Office Action asserts that MathWorks discloses "wherein said automatically generating comprises automatically generating graphical source code corresponding to the plurality of states, wherein the graphical source code comprises a plurality of interconnected nodes which visually indicate functionality of the graphical program", citing p.2-4, and the figure on p.2-7. However, the cited text actually reads:

> The Simulink Model and Stateflow Machine
>
> The Stateflow machine is the collection of Stateflow blocks in a Simulink model. The Simulink model and Stateflow machine work seamlessly together. Running a simulation automatically executes both the Simulink and Stateflow portions of the model.
>
> A Simulink model can consist of combinations of Simulink blocks, toolbox blocks, and Stateflow blocks (Stateflow diagrams). In Stateflow, the chart (Stateflow diagram) consists of a set of graphical (states, transitions, connective junctions, and history junctions) and nongraphical (event, data, and target) objects.
>
> There is a one-to-one correspondence between the Simulink model and the Stateflow machine. Each Stateflow block in the Simulink model is represented in Stateflow by a single chart (Stateflow diagram). Each Stateflow machine has its own object hierarchy. The Stateflow machine is the highest level in the Stateflow hierarchy. The object hierarchy beneath the Stateflow machine consists of combinations of the graphical and nongraphical objects. The data dictionary is the repository for all Stateflow objects.

As may be seen, the cited text describes a Simulink model as consisting of combinations of Simulink blocks, toolbox blocks, and Stateflow blocks (Stateflow diagrams), and a Stateflow diagram as consisting of a set of graphical (states, transitions, connective junctions, and history junctions) and nongraphical (event, data, and target) objects, and further states that there is a one-to-one correspondence between the Simulink model and the Stateflow machine. However, the cited text nowhere discusses or even hints at generating either a Simulink model or a Stateflow diagram *automatically*. The cited figure illustrates a Stateflow diagram within a Simulink model, but again, nowhere discloses automatically generating a graphical program comprising a plurality of interconnected nodes which visually indicate functionality of the graphical program, based on state diagram information. Thus, the cited text and figure fail to teach or suggest this feature of claim 1.

The Office Action further asserts that MathWorks discloses "wherein said automatically generating the graphical program creates the graphical program without any user input specifying the graphical program during said creating", citing p.1-6, specifically:

> By default, an untitled Simulink model with an untitled, empty Stateflow block is created for you when you open the Stateflow model window. You can either start with the default empty model or copy the untitled Stateflow block into any Simulink model to include a Stateflow diagram in an existing Simulink model.

Applicant respectfully notes that as described in the cited text, and shown in the related figure, the cited default Simulink model and Stateflow block are both *empty*, and thus do not include any graphical source code, i.e., a plurality of interconnected nodes, corresponding to a plurality of states, as claimed. Moreover, the MathWorks reference repeatedly describes both Simulink models and Stateflow diagrams (i.e., non-empty ones) as being created *manually*, i.e., in response to user input specifying the model and diagram elements, and nowhere describes such creation *automatically* in response to state diagram information. Thus, the cited text fails to teach or suggest this feature of claim 1.

In the Response to Arguments section, the Examiner states that "'automatically generating' is interpreted to refer to an action being performed by either a program or the user...". Applicant respectfully submits that this attempt to redefine "automatically generating" to include manual (via user input) creation is improper and incorrect, and is at odds with the accepted meaning of the term. Applicant has included language in the claim that particularly states that automatically generating the graphical program creates the graphical program *without any user input specifying the graphical program during said creating*, which is in direct contrast with the MathWorks system. In fact, Applicant notes that the Examiner even quotes "Stateflow provides a block that *you* include in a Simulink model" (*emphasis added*), which specifies that *the user* includes the block in the model.

The Examiner also cites the fact that "Stateflow Coder generates...code based on the Stateflow machine" (p.1-4). Applicant notes that the Stateflow machine is defined by the manually created graphical Simulink model, and reminds the Examiner that, as MathWorks states, "There is a one-to-one correspondence between the Simulink model and the Stateflow machine". Moreover, as MathWorks makes clear, the Stateflow Coder is a "Tool for generating highly readable, efficient *C code* from Stateflow diagrams". In other words, the Stateflow Coder automatically generates text-based program code (C code) from a (manually created) graphical Stateflow diagram, which is quite different from automatically generating graphical source code corresponding to the plurality of states, wherein the graphical source code comprises a plurality of interconnected nodes which visually indicate functionality of the graphical program, based on state diagram information, as claimed.

Thus, for at least the reasons provided above, Applicant submits that MathWorks fails to teach or suggest all the features and limitations of claim 1, and so claim 1 and those claims dependent therefrom are patentably distinct and non-obvious over the cited art, and are thus allowable.

Independent claims 23, 25, 26, 29, and 32 include similar novel limitations as claim 1, and so the above arguments apply with equal force to these claims. Thus, for at least the reasons provided above, claims 23, 25, 26, 29, and 32, and those claims

respectively dependent therefrom, are patentably distinct and non-obvious over the cited art, and are thus allowable.

Removal of the section 102 rejection of claims 1, 3-6, 8, 12-23, 25, 26, 28, 29, and 31-54 is respectfully requested.

**Section 103 Rejections**

Claim 35 was rejected under 35 U.S.C. 103(a) as being unpatentable over MathWorks in view of Kodosky et al (US 5,732,277, "Kodosky").

The Office Action admits that MathWorks fails to teach or suggest "wherein the placeholder graphical source code for each state comprises a case in a graphical case structure", but asserts that Kodosky remedies this admitted deficiency of MathWorks, citing col.20:30-49, col.11:43-60, and col.11:44-60.

Applicant has reviewed the cited passages carefully, and submits that the cited portions in no way teach or suggest this feature. For example, col.20:30-49 reads:

> Case (Conditional) Selection Structure
>
> The case (conditional) selection structure of FIG. 36 is similar to the sequence structure in its usage of screen real estate, but it differs in its handling of arcs crossing the border in that each case diagram must use all incoming arcs and generate all outgoing arcs. The usual execution rules apply (all inputs available to start, all outputs generated simultaneously after all execution completes).
>
> There must be exactly one arc that terminates at the case selection structure border itself, acting as a selector for the case. In the simplest case a boolean valued scalar is connected to the selector to select between case 0 (false) and case 1 (true). In the general case a scalar number (or string variable, if it is specified as a discrete list of strings) is connected to the selector.
>
> A special diagram (default case) may be specified to be used in the event that the selector does not lie in the range of specified cases. An arc originating at the case selection structure border may be used a a ready signal indicating that the selected case has completed execution.

Col.11:43-60 reads:

> The conditional structure, a graphical representation of which is shown in FIG. 10, is similar in appearance to the sequence structure in its use of

screen space, but it differs in its handling of signal paths crossing its border in that in each case a diagram may use any subset of incoming signal paths, but must produce all outgoing signal paths. In accordance with data-flow principles, all inputs must be available in order to start execution. Furthermore, all outputs are generated after execution is completed.

There must be a signal path that terminates at the case-selection terminal on the structure border. In the simplest case, a boolean-valued scaler is connected to the selector to select between case 0 (FALSE) and case 1 (TRUE). In the general case, a scaler number is connected to the selector. A special diagram (default case) may be specified to be used in the event that the selector does not lie in the range of specified cases.

Applicant notes that the text describes embodiments of a graphical case (conditional) selection structure, also referred to as a graphical case structure, or simply conditional structure. Nowhere is the graphical case structure described as including cases comprising placeholder graphical source code for each state (of the second one or more states). Rather, Kodosky's graphical case structure is a general purpose graphical case structure for general use in graphical programming, where the graphical case structure functionality corresponds to that of text-based case/switch statements, well known in the art of programming. Thus, this cited text fails to disclose this feature of claim 35.

Applicant further submits that no proper motivation to combine MathWorks and Kodosky has been provided from the references themselves. The only motivation to combine MathWorks and Kodosky suggest by the Examiner is "to include a case structure so that a menu list of alternatives on the screen for a user to choose from is available". Applicant respectfully notes that Kodosky's graphical case structure is for general programmatic case selection within a graphical program, not for presenting user-selectable menu options to a user, and although it may be possible to generate such a graphical user interface using Kodosky's graphical case structure, such functionality is not described or even hinted at in Kodosky. Nowhere does Kodosky mention or hint at the desirability of an automatically generated graphical program that includes a graphical case structure with placeholder graphical source code for each state comprising cases in the graphical case structure. Nor does MathWorks suggest the desirability of such an

12

automatically generated graphical case structure with placeholder graphical source code for each state comprising cases in the graphical case structure. Applicant further notes that neither reference discloses or even hints at the desirability of automatically generating such a graphical case structure. Applicant further submits that the suggested motivation is not germane to the specific feature claimed, since it is not directed to a graphical case structure as claimed, specifically, "wherein the placeholder graphical source code for each state comprises a case in a graphical case structure". Thus, Applicant submits that no proper motivate on to combine has been provided, and thus that the attempted combination of MathWorks and Kodosky is not available to make a prima facie case of obviousness.

Moreover, even were MathWorks and Kodosky properly combinable, which Applicant argues they are not, the resulting combination would still not produce Applicant's invention as claimed, as argued at length above.

Thus, for at least the reasons provided above, MathWorks and Kodosky, taken singly or in combination, fail to teach or suggest all the features of claim 35, and so claim 35 is patentably distinct and non-obvious over the cited art and is thus allowable.

Moreover, Applicant further submits that since independent claim 32 was shown above to be allowable, claim 35, dependent from claim 32, is similarly allowable, for at least the reasons provided above.

Removal of the section 103 rejection of claim 35 is earnestly requested.


Applicant also asserts that numerous ones of the dependent claims recite further distinctions over the cited art. However, since the independent claims have been shown to be patentably distinct, a further discussion of the dependent claims is not necessary at this time.

## CONCLUSION

Applicant submits the application is in condition for allowance, and an early notice to that effect is requested.

If any extensions of time (under 37 C.F.R. § 1.136) are necessary to prevent the above referenced application(s) from becoming abandoned, Applicant(s) hereby petition for such extensions. If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert & Goetzel PC Deposit Account No. 50-1505/5150-45900/JCH.

Also enclosed herewith are the following items:

☒ Return Receipt Postcard

☒ Terminal Disclaimer to Obviate a Double Patenting Rejection
   Over a Pending Second Application

☒ Terminal Disclaimer to Obviate a Double Patenting Rejection Over a Prior Patent

☒ Power of Attorney by Assignee and Revocation of Previous Powers

Respectfully submitted,

Jeffrey C. Hood
Reg. No. 35,198
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert & Goetzel PC
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8800
Date: _8/22/2006_ JCH/MSW

14